Tactile Sensing Report: Magnetic and Resistive Based Sensor Design for Force Prediction in Three-Dimensional Space

By: Cameron Weigel

California Polytechnic University, Pomona

ME 5741

March 15, 2022

Abstract

A wireless tactile sensor using magnetic and resistive based sensing techniques with the goal to accurately measure a force between .005 N and 2N as well as the direction of the force in 3 dimensional space was prototyped. The application of the sensor was to integrate these concepts and sensor types into a sensorized splint to better understand what forces are being imparted on an injured joint. With hobby-grade sensors, it was difficult to accurately predict an applied force, especially when low forces were applied. Some reasons for the inaccuracy were sensor hysteresis, anisotropic properties of the material embedded with sensors, insufficient calibration and imperfect filtering techniques. Overall, the sensor design was successful in combining multiple sensors into a solid-state sensing "finger", with the caveat that more investment, both time and money, would lead to a sensor that can achieve the requirements presented.

Introduction

Tactile sensing can be viewed as a sensor that senses via physical interaction with its environment, and as such there are many applications involving tactile sensors. Additionally, there are many different types of tactile sensors and sensing methods. A few types of sensing and transduction methods are: capacitive, resistive, magnetic, piezoelectric, and optical sensors, with many other more exotic methods available. This report will focus on two methods of sensing, resistive force sensors, which use a change in resistance of a circuit by applying a force to a material. And, magnetic-based Hall Effect sensors, which use the Hall Effect, a change in voltage caused by the presence of a magnetic field. We will discuss sensing and how it relates to rehabilitation robotics, in particular, sensorized splinting methods. Sensorized splinting is still a very underdeveloped field but has the potential for improvements in patient's pain levels and recovery speed. Utilizing sensorized splints, medical professionals could understand at a deeper level, the forces imparted on an injured joint, and in-turn, have an empirical measurement to relate recovery speed to forces applied to an injured joint. Furthermore, if such a dataset was built, medical professionals could optimize splinting methods, which could further increase recovery speed and pain management. Figure 1 describes a magnetic based sensor developed for sensorized splinting of the wrist:

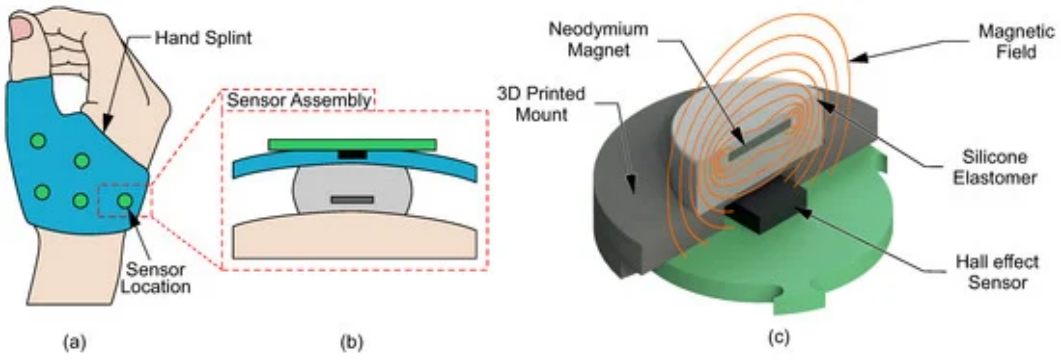

Figure 1: Jones D, Wang L, Ghanbari A, Vardakastani V, Kedgley AE, Gardiner MD, Vincent TL, Culmer PR, Alazmani A. "Design and Evaluation of Magnetic Hall Effect Tactile Sensors for Use in Sensorized Splints." Sensors. 2020; 20(4):1123

In this paper, a sensor was designed and built fusing magnetic and resistive sensors in a silicone medium to create a solid-state sensor. The sensor not only detects forces perpendicular to the sensors, but also with the construction of a three-axis magnetometer, the sensor can detect forces in three dimensions based on the displacement of a magnet embedded in the silicone medium. The purpose of this report is to cover all aspects of the system requirements, system design, manufacturing methods, and final analysis and results of the tactile sensor.

System Design

The initial design of the system was inspired by other tactile sensors developed using similar sensing methods. The sensor itself is composed of several different media; the sensing elements, 3D printed mounting brackets and a solid silicone mold. Figure 2 shows a working prototype of the sensor and its computing platform.

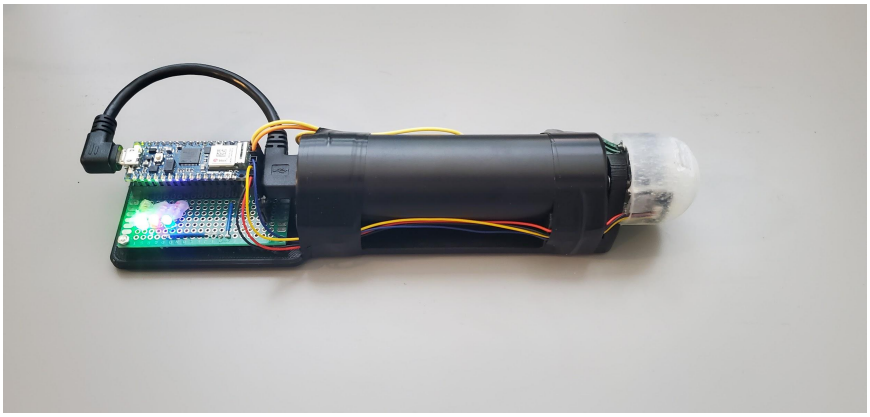

Figure 2: Prototyped sensor featuring a solid-state silicone sensor with peripherals such as battery and light-based haptic feedback

The primary sensor is the Infineon TLV493D-A1B6 3D Magnetic Sensor (See appendix 3.); for convenience, this sensor has been broken out onto a PCB constructed by Adafruit. Some key characteristics of this sensor are its operating range in all three axes, its resolution, noise, and its power consumption. Listed in the datasheet, as well as on the Adafruit product page, the TLV493D has a detectable operating range of -130mT to +130mT. Based on magnet size and position relative to the sensor, one can achieve a precise measurement with minimal displacement of the magnet and in the opposite case, if a wide range of movement is needed, this sensor can accommodate that as well. Experimental data was collected to profile the noise within the sensor, five iterations of calibration under no displacement, with fifty points per iteration, a noise variance of .01 mT was recorded. The datasheet lists noise at 0.1mTRMS, directly correlating with the .01mT variance (or 0.1mT standard deviation) recorded at zero displacement. The TLV493D has the added benefit of having an on-board thermistor for temperature measurement, if the application requires it. If the temperature sensor is not needed, it can be turned off to save power. In this application, the temperature sensor is not used since the sensor is encased in silicone, which would lead to inaccuracies in temperature measurement, additionally, the use-case for a sensorized splint does not immediately warrant a temperature measurement. With the temperature sensor disabled, the sensor consumes, on average, 80 μA in low power mode, and 10 μA in ultra low power mode. At an operating voltage of 3.3V, the sensor consumes on average 2.64mWh of power in low power mode. One key factor in wireless applications is power consumption, by minimizing power consumption, the device is allowed to become much smaller because the device can utilize a smaller battery. In many cases, the largest part of a device, especially one such as this tactile sensor, the battery accounts for the most space and weight. Weight must also be a key factor when designing a sensorized splint, unnecessary, localized weight may lead to uneven pressure applied to the injured joint which could lead to a slower healing process or potential re-injury. Availability of this sensor was also a critical factor in the decision to use this sensor. With a convenient breakout board, low cost, and being in-stock on Adafruit's website, this board was an easy choice.

The auxiliary sensor included in this design was a round force sensitive resistor, this resistor was used in conjunction with the magnetometer to provide redundancy and sensor agreement. The force sensitive resistor used was the Interlink 402 (See appendix 4.) which is also available through purchase on Adafruit or Digikey. The primary factors in deciding to use

this sensor were its size, low cost, availability, and operating range. The sensor's footprint fit within the footprint of the 3-axis magnetometer which provided a straightforward and relatively low-profile sensor stackup. Fixed via superglue to the mounting plate above the magnetometer, the force sensitive resistor lies on the same axis as the embedded magnet, which allows for an easy correlation between the two sensor readings. Experimental data for the force sensitive resistor showed that the minimum readable force applied to the sensor was around 20 grams, while this is higher than the minimum requirement of 6 grams, the force sensitive resistor was still useful as a sensor-agreement measurement for values above 20 grams. For lower values, only the magnetometer sensor readings were considered. Unfortunately, these low-cost force sensitive resistors are known for having relatively high inaccuracies, in future designs, a load-cell or strain gauge should replace the force sensitive resistor.

The computing platform used for this project was the Arduino Nano 33 IOT microcontroller development board (See appendix 1.). This board was chosen for several reasons, its size, available pins, a high clock speed, a 12 bit ADC, several pins designated for various communication protocols, 256kB of flash memory, and a wireless communication module that is both bluetooth and wifi enabled. The Nano 33 IOT has a small footprint of only 45mm x 18mm, which is convenient for the initial prototype being handheld. The main processor onboard is the 32 bit ATSAMD21G18 ARM® Cortex®-M0+. Key metrics to this processor are its 48MHz operating frequency, 256Kb of flash memory, UART/USART, SPI or I2C, three 16-bit timer/counters, 32-bit Real-Time Clock and calendar, 20 PWM channels, one 14-channel 12-bit ADC, one 10-bit DAC, and a Full Speed USB Device and embedded Host. Since the 3-axis magnetometer requires an I2C communication bus and the force sensitive resistor requires an ADC, this microcontroller was an excellent choice for this project. Additionally, since the microcontroller can be programmed to use 12 bits for the ADC, in theory, it has double the precision of a 10 bit ADC, which is useful for the force sensitive resistor. The Arduino IDE and Arduino's library ecosystem were used for programming the microcontroller. The ArduinoBLE.h library was used for accessing the onboard bluetooth features and the Tlv493d.h library was used for easily accessing the 3 axis magnetometer via I2C. Additionally, a custom struct was developed for each axis of the magnetometer and several helper functions were written for filtering and calibration.

The rest of the sensor design consists of 3D printed brackets that sandwich the magnetometer PCB which allows for a convenient sensor stackup and a method of mounting the entire sensor on its case. This sensor stackup was then encased in a Shore 37A silicone (See appendix 6.) via a 3D printed two-part mold. The mold was necessary for two reasons, encapsulating the magnet in a precise location, and in compliance with other sensorized splint designs, the silicone provides a soft and safe contact medium with skin.

A perfboard was used to layout the electrical components required for the sensor. Using a 60mm x 40mm sized perfboard with insulated contacts, components were placed and soldered together in a design that minimized size, and complexity. Additionally, headers were used to allow reuse of the main sensor unit and the microcontroller in any future design. Aside from the headers for both the sensor unit and the microcontroller, three LEDs, one red, one blue and one green were used to provide visual feedback on the force magnitude and direction applied to the sensor. Complying with most standard models of the cartesian axes, the red LED relates to the X-axis, the green LED to the Y-axis, and the blue LED to the Z-axis. Since perfboard is double-sided, the underside of the board contained all the necessary connections to the different electrical components which allowed for a clean layout of the components visible to the user.

In addition to the sensor and perfboard design, a minimal but functional 3D printed case was designed to fix the components together. Standoffs, countersunk features and a battery cage were designed into the case to easily fit all components on the board. Since all components were printed from the same 3D printer, location, and features of size were practically nominal. (include image) For an initial prototype, size was not a major consideration, before optimization should take place, a functioning device must be created. For that reason, this design included a larger, off-the-shelf USB power bank. Still, the size and shape allows the device to be hand-held.

Materials and Manufacturing

Manufacturing methods were limited based on the time given to complete the sensor and a small budget. Most components were off-the-shelf components that could be purchased for a few dollars each, and any designed components could either be 3D printed, or prototyped on a perfboard. In order to incorporate the two sensors into a solid-state sensor, a 3D printed two-part mold was designed for pouring silicone over the sensor stackup. The mold had two stages, the first stage was poured from the top of the mold to encapsulate the sensor stackup. The second

stage involved removing the insert from the side of the mold that created a cavity for the magnet. Once removed, the magnet was then placed inside the cavity and silicone was poured from the side. (See Figure 3)

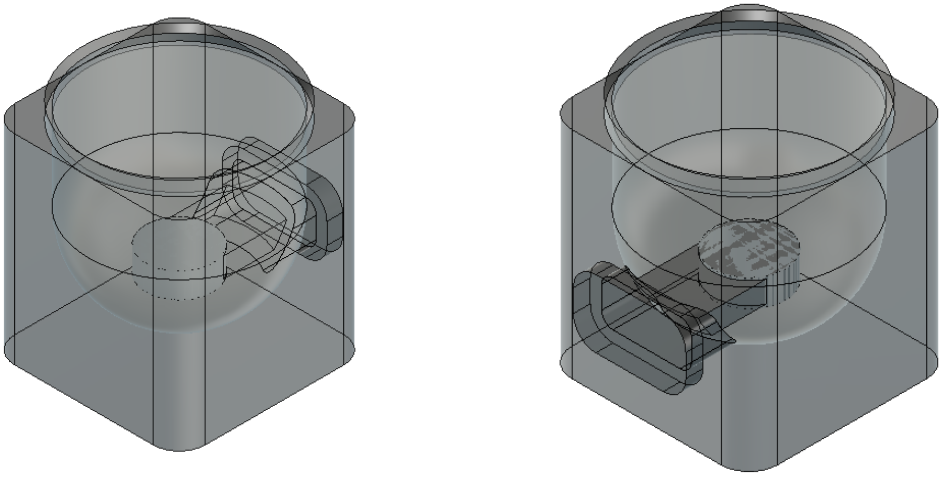

Figure 3: Two part mold for casting silicone around sensors

The mold was designed to minimize size while still fully encasing the sensors. Since these sensors were hobby-grade, their size was significantly larger than what was desired, which led to an oversized silicone sensor.

The electrical circuit was designed on a protoboard with modularity and reusability in mind. In order to accomplish that goal, headers were soldered onto the protoboard so the microcontroller and sensor pinouts could easily be removed so that the case could easily be redesigned. The magnetometer was connected to the microcontroller's I2C communication bus and the force resistive sensor was connected to the ADC with a 5.6k Ohm resistor bridging the ADC voltage in and ground. For the visual feedback provided by the LEDs, the LEDs were connected to PWM pins on the microcontroller with a 220 Ohm resistor across each LED ground line. PWM pins were used in order to control the brightness of the LEDs and the resistor minimized current through the LEDs so as to not burn them out at max duty cycle. Figure 4 shows the perfboard circuitry without the microcontroller or sensors attached. (Include circuit diagram)

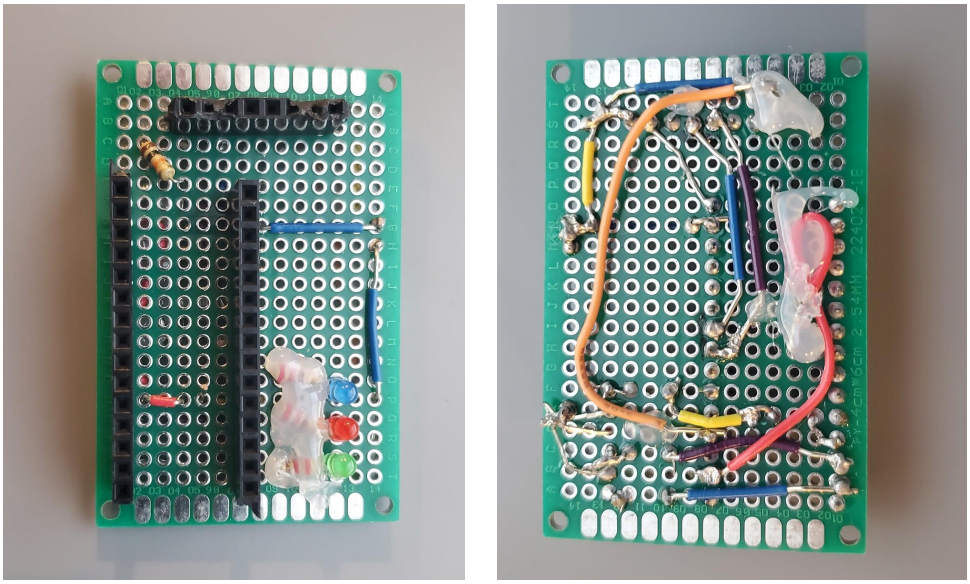

Figure 4: Perfboard used to organize all connections to microcontroller, sensors, and LEDs

Analysis and Results

The final prototype had some design flaws that led to inaccuracies in the predicted force. Sensor noise, anisotropic properties of the silicone mold, position of the magnet, and limitations of the force sensitive resistor, improper filtering and interpolation methods all led to an incomplete force-predictive model.

As discussed previously, a sensor noise RMS of 0.1 mT became a serious issue at low force values due to the solid-state nature of the sensor. With low force values, the sensor readings deviated only very slightly from the no-force value. At these low force values, noise contributed to 60% of the variation in the data from point to point, making it nearly impossible to detect forces below 12 grams. This was primarily caused by the hardness of the silicone, a shore 37A silicone was used which led to minimal displacement when applying low forces. A softer silicone, possibly shore 20A, low force readings might have been more detectable. Additionally, there was a nonlinear relationship between force and displacement, as the force reached certain thresholds, the required force to displace the magnet increased, meaning a model that predicts force through displacement of the magnet must be nonlinear. An attempt was made to correct for this nonlinearity by including scaling factors of the force reading based on the range of the sensor measurement.

Several techniques of filtering were applied in an attempt to minimize the effects of noise and nonlinearity, but all proved insufficient in achieving the requirements of the system. For force predictions, many filtering methods were tested. Moving average filters, exponential filters, low-pass filters, Kalman filtering, thresholding, point interpolation, and combinations of those filters were tested. Moving average filters incorporated too much noise and would not stabilize at low values. Additionally, exponential filters were too sensitive to the noise or would lead to hysteresis based on their weights. Kalman filtering is a new technique to me and because of that, I did not want to implement a filter without fully understanding it, regardless, my Kalman filter did not result in a more stable measurement. In the end, a combination of an exponential filtering and thresholding were utilized to most accurately predict forces. The point to point equation is listed below.

$$y_n = w * x_n + (1 - w) * y_{n-1}$$

$$Where\ y_n\ and\ y_{n-1}\ denotes\ the\ predicted\ force\ value\ and\ the\ previous\ force\ value$$

$$x_n\ is\ the\ current\ data\ point$$

$$And\ w\ is\ the\ weights,\ w = .05\ for\ this\ application$$

Unfortunately, the most accurate model developed in time had inaccuracies of up to 10 grams, which was particularly inaccurate at low force readings.

Calibration methods incorporated a larger moving average window of 50 samples but were susceptible to the same issues. Additionally, based on the position of the sensor in 3D space, calibration values would change and would therefore not be accurate if the sensor is then tested in a different position. Another major factor that contributed to calibration inaccuracy was the design of the sensor mounting features. Since the sensor case was 3D printed, there was some compliance in the structure as higher forces were applied to it, this led to unintended movement of the sensor stackup, leading to further obfuscation of the force prediction.

Conclusion

As an initial prototype, the design and function of the sensor was extremely valuable from a learning standpoint. Although the sensor did not fully satisfy all the requirements, for a low-cost implementation using hobby sensors, the granularity and robustness of the predicted force values and force directions I found quite impressive. If another iteration of the project was

to be designed, there are several major changes I would make. Starting with the mechanical design, redesigning the sensor mold with both a smaller magnet and a magnet and a softer material to allow more deformation of the silicone to utilize the entire resolution of the magnetometer. By changing both the magnet and the silicone, the noise present in the system would account for a smaller percentage of the data, leading to a more accurate force prediction model, especially at lower force values. Incorporating accelerometer and gyroscope data into the calibration model so that changes in position of the entire sensor would not affect sensor readings. Using a redundancy method other than a force sensitive resistor since most force sensitive resistors are not very accurate, replacing it with a load cell or strain gauge would be ideal. Finally, reducing the battery size and designing a custom PCB to reduce the overall size and weight of the sensor in an effort to make the sensor integrable into a splint. From a software perspective, having a better understanding of filtering techniques and interpolation methods could lead to a more accurate force prediction model.

Appendix

1. Arduino Nano 33 IOT Datasheet
2. ATSAMD21G18 Product Page
3. Adafruit Triple Axis-Magentometer
4. Adafruit Round Force-Sensitive Resistor
5. TLV493 Datasheet
6. Smooth-On SORTA-Clear 37 Silicone

Code

```
#include <ArduinoBLE.h>
#include <Tlv493d.h>

struct magAxis{
  float val = 0;
  float sum = 0;
  float readings[50];
  float averaged = 0;
  float cal = 1.0;
  float cur_pt = 0.0;
  float prev_pt = 0.0;
```

```
  float w = 0.05;
  float window[10];
  float smoothed = 0.0;
  float smoothed_prev = 0.0;
  float variance = 0.0;
  float kalman = 0;
  long fsr = 0;

  float low_pass_0 = 0.0;
  float low_pass_1 = 0.0;
};


int k = 0;
int fsrPin = 0;
int fsrReading;
int i = 0;
unsigned long myTime;

String address;

BLEService magnetometerService("19B10010-E8F2-537E-4F6C-D104768A1214"); // create
service


// create axial characteristics and allow remote device to get notifications
BLEFloatCharacteristic xCharacteristic("MagnetometerXReading", BLERead |
BLENotify);
BLEFloatCharacteristic yCharacteristic("MagnetometerYReading", BLERead |
BLENotify);
BLEFloatCharacteristic zCharacteristic("MagnetometerZReading", BLERead |
BLENotify);


// Tlv493d Opject
Tlv493d Tlv493dMagnetic3DSensor = Tlv493d();


int z_led = 9;
int z_led_val = 0;

int x_led = 10;
int x_led_val = 0;

int y_led = 11;
int y_led_val = 0;
```

```
bool find_avg = true;

magAxis zAxis;
magAxis xAxis;
magAxis yAxis;


float interpolate(float x) {
  if ( x < .08) {
    return 0.0;
  }
  if (x >= .12  && x <=.15) {
    return x*128.0;
  }
  if (x > .15 && x <= .2) {
    return x*166.0;
  }
  if (x >.2 && x <= .3) {
    return x*231.0;
  }
  if (x > .3) {
    return x*333.0;
  }
};

void setup() {
  pinMode(z_led, OUTPUT);
  pinMode(x_led, OUTPUT);
  pinMode(y_led, OUTPUT);

  Serial.begin(9600);
  Serial.println("X:,Y:,Z:,");
  //while (!Serial);
  Tlv493dMagnetic3DSensor.begin();
  // begin initialization
  if (!BLE.begin()) {
    Serial.println("starting BLE failed!");

    while (1);
  }

  // set the local name peripheral advertises
  BLE.setLocalName("MagnetometerReading");
  // set the UUID for the service this peripheral advertises:
  BLE.setAdvertisedService(magnetometerService);

  // add the characteristics to the service
```

```
  magnetometerService.addCharacteristic(xCharacteristic);
  magnetometerService.addCharacteristic(yCharacteristic);
  magnetometerService.addCharacteristic(zCharacteristic);

  // add the service
  BLE.addService(magnetometerService);

  xCharacteristic.writeValue(0);
  yCharacteristic.writeValue(0);
  zCharacteristic.writeValue(0);

  // start advertising
  BLE.advertise();

  //Serial.println("Bluetooth device active, waiting for connections...");
  address = BLE.address();
  //Serial.println(address);
}

void loop() {
  // poll for BLE events
  myTime = millis();
  BLE.poll();
  Tlv493dMagnetic3DSensor.updateData();

  zAxis.val = Tlv493dMagnetic3DSensor.getZ();
  xAxis.val = Tlv493dMagnetic3DSensor.getX();
  yAxis.val = Tlv493dMagnetic3DSensor.getY();
  analogReadResolution(12);

  fsrReading = analogRead(fsrPin);

  //Serial.println(fsrReading);
  if (myTime < 10000)
  {
    //zAxis.sum = zAxis.sum - zAxis.readings[i];
    zAxis.readings[i] = zAxis.val;
    zAxis.window[k] = zAxis.val;
    //zAxis.sum = zAxis.sum + zAxis.val;

    //xAxis.sum = xAxis.sum - xAxis.readings[i];
    xAxis.readings[i] = xAxis.val;
    //xAxis.sum = xAxis.sum + xAxis.val;

    //yAxis.sum = yAxis.sum - yAxis.readings[i];
    yAxis.readings[i] = yAxis.val;
    //yAxis.sum = yAxis.sum + yAxis.val;
```

```
  //zAxis.averaged = (zAxis.sum / 20);
  //xAxis.averaged = (zAxis.sum / 20);
  //yAxis.averaged = (zAxis.sum / 20);

  k = (k+1) % 10;
  i = (i+1) % 50;
  delay(20);
}
else if (myTime > 10000 && find_avg == true)
{
  for (int j = 0; j<50; j++){
    zAxis.sum = zAxis.sum + zAxis.readings[j];
    xAxis.sum = xAxis.sum + xAxis.readings[j];
    yAxis.sum = yAxis.sum + yAxis.readings[j];
  }

  zAxis.averaged = (zAxis.sum / 50);
  xAxis.averaged = (xAxis.sum / 50);
  yAxis.averaged = (yAxis.sum / 50);

  find_avg = false;

  zAxis.sum = 0;

}
else
{
  zAxis.low_pass_1 =zAxis.w*zAxis.val + (1-zAxis.w)*zAxis.low_pass_0;
  zAxis.low_pass_0 = zAxis.low_pass_1;
  xAxis.window[k] = xAxis.val;
  yAxis.window[k] = yAxis.val;
  zAxis.window[k] = zAxis.low_pass_1;
  k = (k+1) % 10;

  zAxis.low_pass_1 =zAxis.w*zAxis.val + (1-zAxis.w)*zAxis.low_pass_0;
  if (abs(zAxis.low_pass_1 -  zAxis.averaged) > .075) {
    //Serial.println("Force has changed");
    Serial.println(interpolate(zAxis.low_pass_1 - zAxis.averaged));
    //Serial.println(zAxis.low_pass_1 - zAxis.averaged);
  }
  //Serial.println(zAxis.low_pass_1 - zAxis.low_pass_0,4);
  zAxis.low_pass_0 = zAxis.low_pass_1;

  //Serial.println(zAxis.low_pass_1*10);
```

```
    if (k == 0) {
      for (int i = 0; i<10; i++){
        xAxis.sum += xAxis.window[i];
        yAxis.sum += yAxis.window[i];
        zAxis.sum += zAxis.window[i];
      }
      xAxis.smoothed = xAxis.sum/10.0;
      xAxis.sum = 0;

      yAxis.smoothed = yAxis.sum/10.0;
      yAxis.sum = 0;

      zAxis.smoothed = zAxis.sum/10.0;
      zAxis.sum = 0;
      zAxis.cal = (zAxis.smoothed);
      //Serial.println(zAxis.cal);
      //Serial.println(zAxis.low_pass_1);
    }


    // Write to LED Pin
    z_led_val = map(zAxis.smoothed*100,5240, 5440, 0, 255);
    zAxis.fsr = map(fsrReading, 0, 4096, 0, 255);
    analogWrite(z_led, z_led_val);

    // Write to LED Pin
    x_led_val = map(xAxis.smoothed*100, -550, -1300, 0, 255);
    analogWrite(x_led, x_led_val);

    // Write to LED Pin
    y_led_val = map(yAxis.smoothed*100, -200, 600, 0, 255);
    analogWrite(y_led, y_led_val);




    delay(100);
//    Serial.print("Z:");
//    Serial.print(z_led_val);
//    Serial.print(",");
//    Serial.print("X:");
//    Serial.print(x_led_val);
//    Serial.print(",");
```

```
//    Serial.print("Y:");
//    Serial.println(y_led_val);

    //Serial.println(zAxis.cur_pt);
    //Serial.println(z_led_val);


//    Serial.println(zAxis.fsr);



    xCharacteristic.writeValue(Tlv493dMagnetic3DSensor.getX());
    yCharacteristic.writeValue(Tlv493dMagnetic3DSensor.getY());
    zCharacteristic.writeValue(Tlv493dMagnetic3DSensor.getZ());
    //delay(250);
  }

}
```